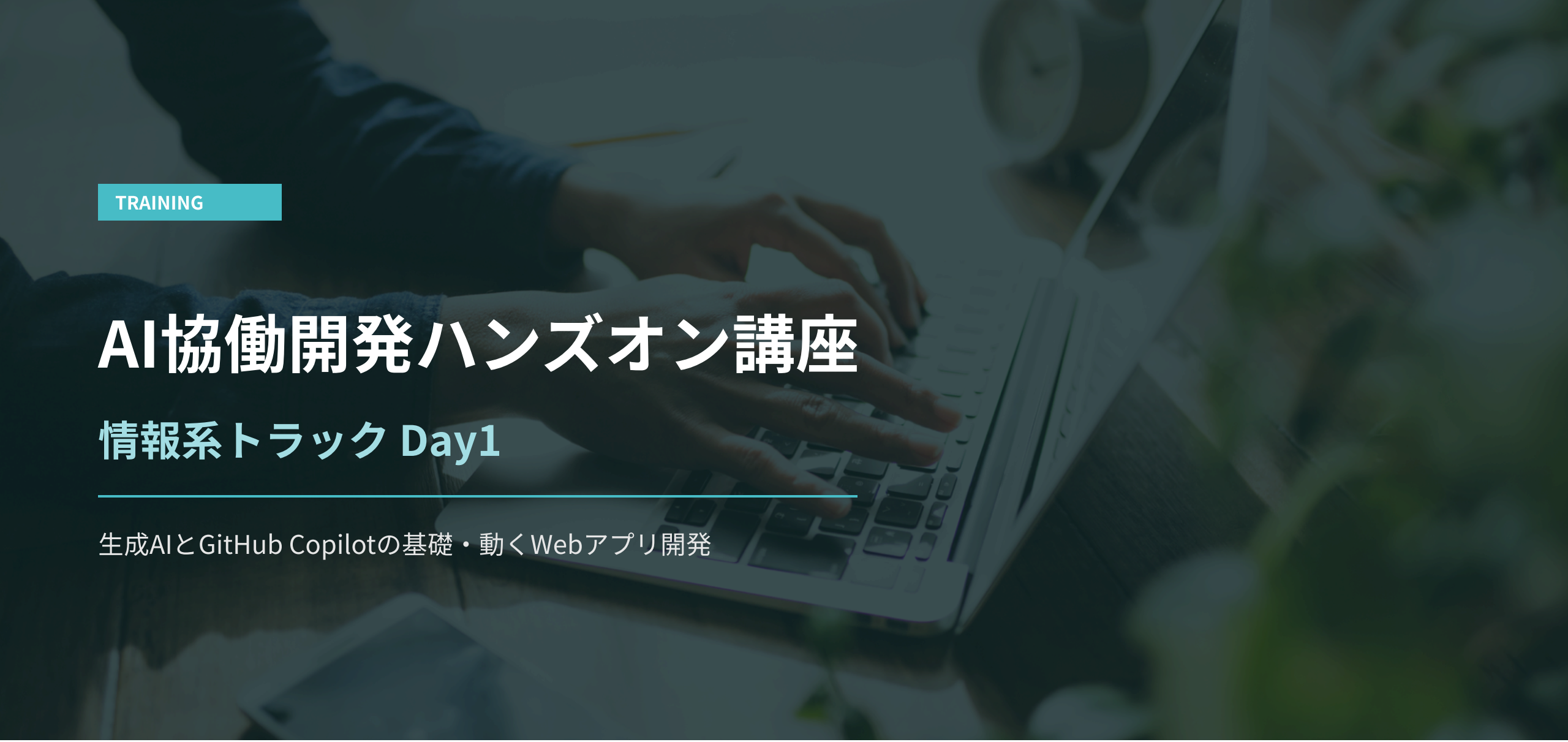


TRAINING

AI協働開発ハンズオン講座

情報系トラック Day1

生成AIとGitHub Copilotの基礎・動くWebアプリ開発

Givery

本資料の二次転載禁止について

本資料の二次転載を禁止します。

本資料は株式会社ギブリーが作成した著作物です。受講者ご本人の学習および業務での参照に限りご利用いただけます。社外への配布、SNS等への掲載、第三者への共有はご遠慮ください。

研修中の画面撮影・録画も、資料が写り込む範囲ではお控えください。

本日のゴール

今日は環境を動かし、AIの守備範囲を説明できる状態にします

VSCodeでCopilotが応答する
貸与PCのVSCodeでチャットに日本語で質問を投げ、日本語で返答が返るところまで確認します。今日いちばんの山はここです。

補完・チャット・エージェントを区別できる
3つを機能名ではなく、任せる作業の大きさで説明できるようにします。

AIに任せない範囲を線引きできる
人が判断すべきところを午前に決めます。客先情報の扱いは午後にまとめて決めます。

3つとも満点にする必要はありません。1番目だけは全員そろえます。

3日間の位置関係

3日間は、作る側から検証する側までを一続きに扱います

Day1

Copilotの基礎操作と、仕様から動くWebアプリを1本。終わったとき、ブラウザで動くHTMLが手元に1つ残ります。

Day2

配布の作業記録アプリへの機能追加、モデルの使い分け、copilot-instructions.md。既存コードを読ませて直せる状態にします。

Day3

単体テストとE2E、セキュリティ観点、実技課題と修了。仕様からテストを設計できる状態にします。

各日8時間、Day間はおよそ1週間あきます。日程は第1回・第2回と呼びます。

今日1日の流れ

手を動かす時間は合計で約240分。座って聞く時間より長くなります

環境セットアップ [	AI駆動開発の世界 [60min]	昼休憩	基本操作とプロンプト設計	仕様からアプリ、壊して直す [	振り返り [20min]
VSCode、GitHubアカウント、Copilotの招待、疎通確認まで。こ	LLMの動き方と、補完・チャット・エージェントの違い。手を止めて聞	長さと入る位置は当日ご案内します。	チャット、スラッシュコマンド、エージェントモードを自分の指で触り	作業記録アプリを題材に、生成と機能追加とデバグまで。今日の山場で	今日の往復を数えて、明日からの持ち帰りを決めます。

各ブロックの間に[10min]の休憩が入ります。昼休憩の長さと入る位置は（要確認）です。

ここから手を動かします

ここから110分は、説明より作業の時間です

ここまでにやったこと

今日のゴール3つと、3日間の並び、今日1日の時間の使い方を共有しました。ここまでは全員が座って聞く時間です。

これからやること

ここから[110min]、手を動かします。VSCodeを開き、GitHubアカウントを作り、Copilotの招待を承諾して、チャットが日本語で返るところまで全員そろえます。

そのあとどこで効くか

午後の演習は、この状態を前提に始まります。1つでも欠けていると、午後は自分の端末で何も試せません。だからここで全部そろえます。

01

環境セットアップ

VSCodeとGitHub Copilotを、このPCで動く状態にする

今日使う道具

コードはVSCodeで書き、動かす場所はブラウザに固定します

書く	VSCode。1.116以降はCopilot Chatが標準で入っているので、拡張機能の追加インストールはしません。
動かす	ブラウザ。作ったHTMLをそのまま開いて確認します。サーバーは立てません。
渡す	データを読ませるときはファイルをウィンドウへドラッグ＆ドロップします。file:// で開いたページはfetchで自動読込ができないためです。
使わないもの	Python、Node.js、コマンドライン。管理者権限も要りません。

普段のIDEがVisual StudioやIntelliJでも、今日はVSCodeに統一します。考え方はそのまま持ち帰れます。
出典: https://code.visualstudio.com/updates/v1_116

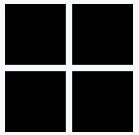
解説

今日触る3つの製品

書く場所と動かす場所を分け、その間にCopilotを置きます

VSCode
コードを書く場所です。
1.116以降はCopilot Chat
が標準で入っています

GitHub Copilot
補完とチャットと
エージェントを出す本体で
す。VSCodeの中で動きます



Microsoft Edge
動かす場所です。保存した
HTMLをダブルクリックして
開きます

出典: Visual Studio Code 1.116 リリースノート https://code.visualstudio.com/updates/v1_116

配布ZIPの中身

配布ZIPは、デスクトップに tpro-work という名前で展開します

中身	何に使うか	いつ触るか
作業記録アプリ/ index.html	日付・作業種別・内容・工数の4列を記録する既存アプリ。今日と翌週の題材です。	午前と午後
作業記録アプリ/ README.txt	今できることと、まだできないことの一覧。	午後
spec5.txt	午後にAIへ貼りつける仕様の5行。手打ちせず、ここからコピーします。	午後
worklog-complete.html	午後に作るアプリの完成版。生成が通らなかった人の差し替え用です。	午後（使う人だけ）
lib フォルダ（3 ファイル）	xlsx.full.min.js、papaparse.min.js、chart.umd.js。合計約1.1MB。	Day2以降

セットアップの5手順

5つの手順は、全員がそろってから次に進みます

1 VSCodeを開く	2 GitHubアカウント	3 招待を承諾	4 VSCodeでサインイン	5 疎通確認
起動とバージョン確認、tpro-work を開くところまで。	会社のメールアドレスで新規作成。ここが一番時間を使います。	講師がこの場で招待を送ります。届いたら承諾します。	チャットパネルからGitHubアカウントで認証します。	チャットが日本語で返ること、補完が出ることを見ます。

> 投影の隅に今のステップ番号を出しておきます。自分がどこにいるか分からなくなったら手を挙げてください。

VSCodeの画面の名前

今日出てくる画面の名前は、この5か所だけです

1. **アクティビティバー**
   左端の縦並びのアイコン。今日はいちばん上のエクスプローラーと、チャットのアイコンだけ使います。
2. **エクスプローラー**
   開いているフォルダの中身。ここに tpro-work の中のファイルが並びます。
3. **エディタ**
   中央。コードを書く場所です。補完の候補は、ここに薄いグレーの文字で出ます。
4. **チャットパネル**
   右側。今日の主役です。サインインもここから始めます。
5. **ステータスバー**
   最下段。今開いているファイルの言語が出ます。補完が出ないときはここを見ます。

ワーク

VSCodeとフォルダの準備

作業場所を tpro-work の1か所に決めてしまいます

どんな場面か

客先の端末を渡された初日にまずやるのは、自分の作業場所を1つ決めることです。今日はデスクトップの tpro-work に、今日つくるファイルを全部集めます。散らばると午後の演習で自分の成果物が見つかりません。

やること

1. スタートメニューから Visual Studio Code を起動します。見当たらない場合は手を挙げてください。
2. ヘルプ、バージョン情報 の順に開き、1.116以降であることを確認します。
3. 配布ZIPをデスクトップに展開し、フォルダ名が tpro-work になっていることを確認します。
4. ファイル、フォルダーを開く から tpro-work を選び、左のエクスプローラーに 作業記録アプリ と lib が並ぶことを確認します。

できあがるもの

成果物はありません。VSCodeのエクスプローラーに tpro-work の中身が見えていれば達成です。

手順1の達成基準

3つとも自分の画面で確認できたら、次に進みます

達成の目安

- タイトルバーかエクスプローラーの最上段に tpro-work と出ている。
- エクスプローラーに 作業記録アプリ フォルダと lib フォルダの2つが見えている。
- バージョン情報に 1.116 以上の数字が出ている。

つまずいたら

- バージョンが1.116より古い。チャットパネルが出ないので、拡張機能から GitHub Copilot Chat を入れます。管理者権限は要りません。
- デスクトップに書き込めない端末。ドキュメントフォルダの直下に tpro-work を作ってください。以降のページのデスクトップを読み替えます。
- フォルダを開かずにファイル単体で開いている。AIが周辺のファイルを読めません。ファイル、フォルダーを開く でやり直します。
- 展開したフォルダが二重になっている。tpro-work の中に tpro-work がある人は、中身を1階層上げます。

手順1の達成基準（続き）

時間が余ったら

- 表示メニューから外観、そして日本語表示の切り替えがどこにあるかを探してみてください。
  英語表示のままでも今日の進行に支障はありません。
- 作業記録アプリ の README.txt を開き、まだできないことの一覧を読んでおいてください。
  午後にここへ機能を足します。

解説

GitHubとCopilotの関係

Copilotの席は、GitHubアカウントに紐づいて配られます

GitHub

アカウントの置き場所です。今日は会社のメールアドレスで新しく作ります

GitHub Copilot

そのアカウントへ席が割り当てられます。招待を承諾すると使えます

ワーク

GitHubアカウント作成

アカウントは会社のメールアドレスで作ります

どんな場面か

Copilotの席は、個人のGitHubアカウントに割り当てられます。今日の招待は会社のメールアドレスに届くので、そのアドレスで作らないと席を受け取れません。午前でいちばん時間を使うのがここです。

やること

1. ブラウザで github.com を開きます。個人アカウントでログイン済みの場合は、先にサインアウトします。
2. Sign up から、会社から支給されているメールアドレスを入力します。個人アドレスでは招待が届きません。
3. パスワードは15文字以上にすると条件で悩まずに通ります。画面に出る要件は当日の表示に従ってください。
4. ユーザー名は世界で重複しない半角英数字。姓のローマ字に数字を足す形で即決してください。
5. 画面のパズル認証を通し、メールに届いた確認コードを入力します。

できあがるもの

成果物はありません。ログイン後の画面右上に自分のユーザー名が出ていれば達成です。

手順2の達成基準

自分のユーザー名が画面に出ていれば、席を受け取る準備は完了です

達成の目安

- github.com の右上のアイコンを押すと、自分のユーザー名が表示される。
- Settings、Emails に、会社から支給されたメールアドレスが登録されている。

つまずいたら

- 確認コードが届かない。迷惑メールフォルダ、次に社内のメールフィルタを見ます。5分待って届かない人は講師が控えて先に進みます。
- ユーザー名が重複して赤字が出る。数字を2桁足せば通ります。ここで悩む時間がもったいないので即決してください。
- パズル認証が重い。回線が混んでいるだけです。閉じずに待ってください。
- 既存の個人アカウントのままログインしている。サインアウトしてから作り直します。

時間が余ったら

- Settings の左メニューから Copilot の項目を開き、今は何も割り当てられていないことを見ておいてください。次の手順で表示が変わります。

ワーク

招待の承諾とサインイン

承諾とサインインで、同じアカウントを使い続けます

どんな場面か

席の割り当ては、招待の承諾とVSCode側のサインインの2段です。ここで別のアカウントが混ざると、承諾したのに使えないという状態になります。研修中いちばん多い事故です。

やること

1. 講師がこのページで招待を一斉送信します。件名に GitHub Copilot を含むメールが届くまで待ちます。数分かかることがあります。
2. 先にブラウザで、さきほど作ったアカウントにログインした状態にします。別アカウントのままだと承諾が別の席に付きます。
3. メール内のボタンから開き、参加を承諾します。

ワーク

招待の承諾とサインイン（続き）

やること

1. VSCodeに戻り、左端のアクティビティバーのチャットアイコン、または 表示 メニューからチャットを開きます。
2. パネル内のサインインを押すとブラウザが開きます。承諾に使ったアカウントかを声に出して確認してから承認します。
3. Visual Studio Code を開きますか のダイアログで 開く を押します。VSCodeの入力欄に文字が打てれば完了です。

できあがるもの

成果物はありません。チャットパネルの入力欄に文字が打てる状態になっていれば達成です。

手順3と4の達成基準

入力欄に文字が打てたら、Copilotが使える状態です

達成の目安

- GitHubの Settings、Copilot に割り当てが表示されている。メニュー名は当日の画面で読み替えます。
- VSCodeのチャットパネルでサインインのボタンが消え、入力欄に文字が打てる。
- VSCode左下のアカウントアイコンに、会社アドレスのアカウントが出ている。

つまずいたら

- 承諾リンクで招待がありませんと出る。ほぼ全部ログインアカウントの取り違えです。サインアウトして入り直します。
- 社内のメールでリンクが書き換えられて開けない。URLをコピーしてブラウザのアドレス欄に貼ります。
- ブラウザからVSCodeへ戻る許可ダイアログで止まっている。開く を押して構いません。
- チャットパネルが出てこない。表示メニューから開き、それでも出なければバージョンを確認します。1.116より古い端末は拡張を入れます。
- 会社のプロキシで認証が通らない。その場では直りません。講師が端末と症状を記録して持ち帰ります。今日は講師機の画面で進みます。

手順3と4の達成基準（続き）

時間が余ったら

- Settings の Copilot で、公開されているコードに一致する提案をブロックする設定がどこにあるかを探しておいてください。Day2で扱います。

ワーク

疎通確認

午後の演習は、この3つが通っている前提で始まります

どんな場面か

1つ欠けたまま午後に入ると、自分の端末では何も試せません。講師が見て回らなくても各自で判定できる形にしてあります。3つ終わった人は次のページの課題へ進んでください。

やること

1. エクスプローラーで 作業記録アプリ の index.html を開きます。
2. チャットパネルに「このファイルは何をするアプリですか。日本語で答えてください」と入力して送ります。
3. tpro-work の中に新しく test.html を作って保存し、エディタで html と打って数秒待ちます。
4. チャット入力欄の下にあるモデル名を押し、一覧が開くことだけ見ます。切り替えはDay2で扱います。

できあがるもの

デスクトップの tpro-work の中に test.html が1つできます。中身は空でも構いません。

疎通確認の達成基準

内容の正確さは問いません。返ってくることだけ確認します

達成の目安

- チャットから日本語のテキストが返ってくる。内容が実際のコードと合っているかは、ここでは問いません。
- test.html で html と打つと、薄いグレーの文字で候補が出る。
- モデル名を押すと、複数の名前が一覧に出る。

つまずいたら

- 返答が英語で返る。文末に 日本語で答えてください を足せば直ります。指示の効き方の実例なので、その場で全体に共有します。
- 補完が出ない。ファイル名が test だけになっていることが大半です。.html を付けて保存し直し、ステータスバーの表示も見ます。
- 返答が返ってこない。回線が混んでいるだけのことが多いので30秒待ってから送り直します。それでも来ない場合は講師に伝えてください。
- 返答が index.html と無関係な内容になる。ファイルを開いた状態で送れているかを確認します。閉じていると中身が渡りません。

疎通確認の達成基準（続き）

時間が余ったら

- 3つ終わった人は次のページの課題A〜Dへ進んでください。待つ必要はありません。
- 同じ質問をもう一度送って、返ってくる文が前と違うことを見ておいてください。次の章でその理由を扱います。

発展

先に終わった人へ

終わった人はここから勝手に進んでください

課題	やること	達成の目安
A	作業記録アプリの index.html をチャットに説明させ、返答の中で実際のコードと違う箇所を探す。	違う箇所を1つ以上、口で言える。
B	チャット入力欄のモデル切替を開き、選べるモデル名を書き出す。	モデル名を3つ以上メモできた。Day2で使います。
C	index.html をブラウザで開き、F12の Console にエラーが出ていないか見る。	Console に赤い行が出ていないことを確認できた。
D	GitHubの Settings、Copilot で、公開コードに一致する提案をブロックする設定の場所を探す。	その設定画面を開けた。

> 分からなくなったら手を挙げてください。周りを手伝ってもらうのは最後の手段にします。

環境が動いたので、中で何が起きているかに移ります

ここまでにやったこと

VSCodeでチャットが日本語で返り、補完が出るところまで全員そろいました。手元に動く環境が1つあります。

これからやること

ここから[60min]は手を止めて聞く時間です。LLMが何を見て答えを出しているのか、AIが自分で動くとはどういうことかを整理します。

そのあとどこで効くか

午後は同じ画面で、この整理を使い分けます。仕組みを知らないまま触ると、返ってきたものを検証できません。

02

AI駆動開発の世界

LLMの動き方、エージェント、Copilotの設計思想を押さえる

LLMが答えを出すまで

LLMは文章を細かく分け、次に続く語を確率で選んでいます

入力を分ける

送った文章はトークンという単位に分かれます。日本語は1文字が1トークン前後で、英語より多く使います。

→

文脈をまとめて読む

過去のやり取りと、開いているファイルの中身が、同じ入力として一緒に渡されます。

→

次の語を確率で選ぶ

学習した分布から、次に来やすい語を選びます。同じ質問で答えが揺れるのはこれが理由です。

→

文章として返す

選び続けた結果が回答です。内容が正しいかを検算する仕組みは入っていません。

> 答えが毎回変わるのは不具合ではありません。仕組み上そうなります。

渡っているものと渡っていないもの

文脈に入っていない情報は、使われません

渡っていないもの

開いていない他のファイル、社内規約、客先の仕様書、実行した結果。ここにある情報は、聞いても答えに使われません。代わりにもっともらしい推測が返ります。

渡っているもの

開いているファイル、選択した範囲、これまでの会話、明示的に添付したファイル。ここに入れた分だけが判断材料になります。

AIは書く作業に強く、正しさの保証は持っていません

	得意な作業	苦手な作業
1	定型的な雛形の生成。入力フォームや一覧表など、書けるが面倒なもの。	客先固有の仕様と社内規約は知りません。渡さない限り出てきません。
2	既存コードの説明とコメント付与。読む手間が減ります。	最新のライブラリ仕様は、学習した時点までしか知りません。
3	命名と型の統一。数十か所の書き換えに強く出ます。	存在しないメソッド名を、自信のある文体で出します。

> 苦手側は工夫で消えるものではありません。仕組み上そうなるので、前提として扱ってください。

自信のある文体で返る誤り

動かすまで、正しいかどうかは分かりません

作業記録アプリで工数の合計を出させたときに返ってきた例です。読みやすく書かれていますが、2行目のメソッドは存在しません。

```
// 返ってきたコード（このままでは動きません）
const total = records.sumBy('hours');
// 配列に sumBy はありません。実行すると TypeError で止まります。

// 実際に動く形
const total = records.reduce((sum, r) => sum + r.hours, 0);
```

> この1行は、読んだだけでは間違いに見えません。ブラウザで開いてConsoleを見た瞬間に分かります。

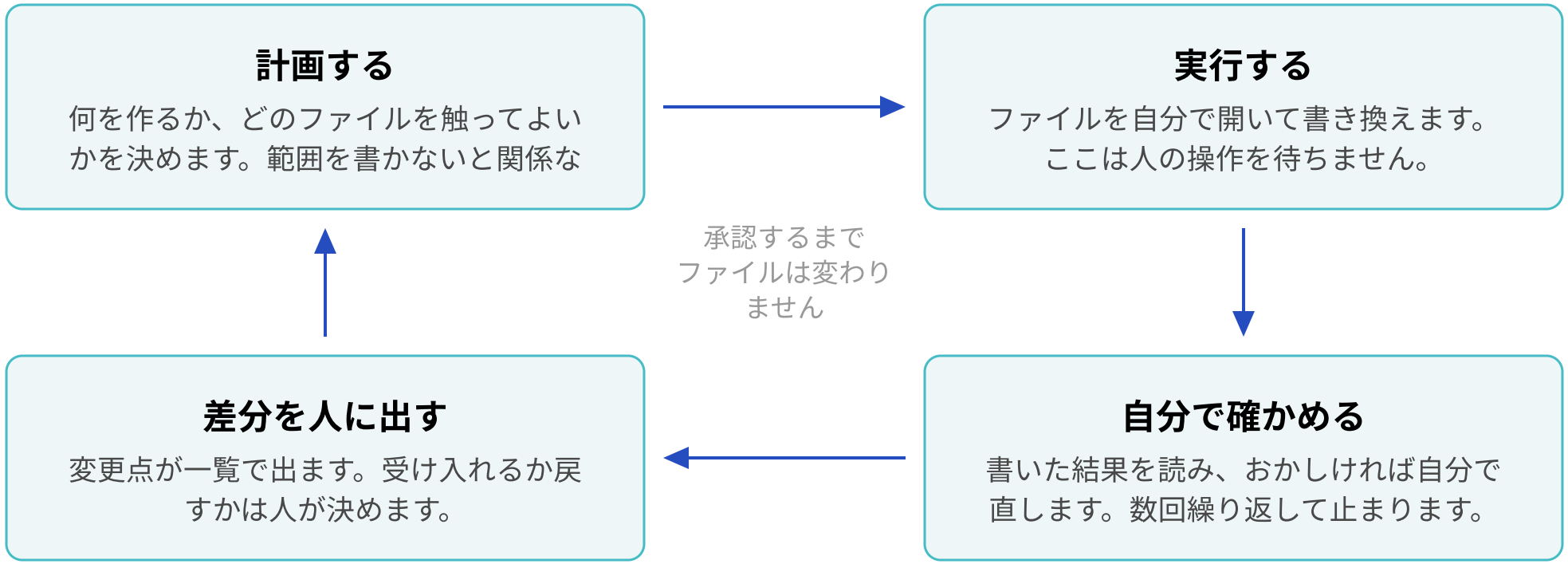
3つは同じAIですが、任せる作業の大きさが違います

機能	任せる範囲	人がすること	いつ使うか
補完	カーソル位置の続きを数行。	採るか消すかを即決する。	書き出しが決まっていて、手を止めたくないとき。
チャット	選択した範囲か1ファイルの相談。	渡す範囲を決め、返答を検証する。	作り方を相談したいとき、既存コードの意味を知りたいとき。
エージェント	複数ファイルの編集とやり直し。	作業前に範囲を指定し、出た差分を読む。	ファイルをまたぐ変更、ゼロから1本つくるとき。

自分でファイルを開いて貼るのがチャット、AIが自分でファイルを開くのがエージェントです。切替の場所は午後に画面で確認します。

エージェントの1周

エージェントは計画と実行と自己修正を、自分で回します



止める場所は2か所です。触る対象が決まった直後と、差分が出た後。それ以外は口を出さなくて構いません。

差分の受け入れ方

読まずに承認した行は、自分が書いた行になります

まとめて受け入れる

差分を読まずに全部承認します。動かなくなったとき、どこが変わったのか誰も分かりません。今日はGitを使わないので、戻す手段は限られます。

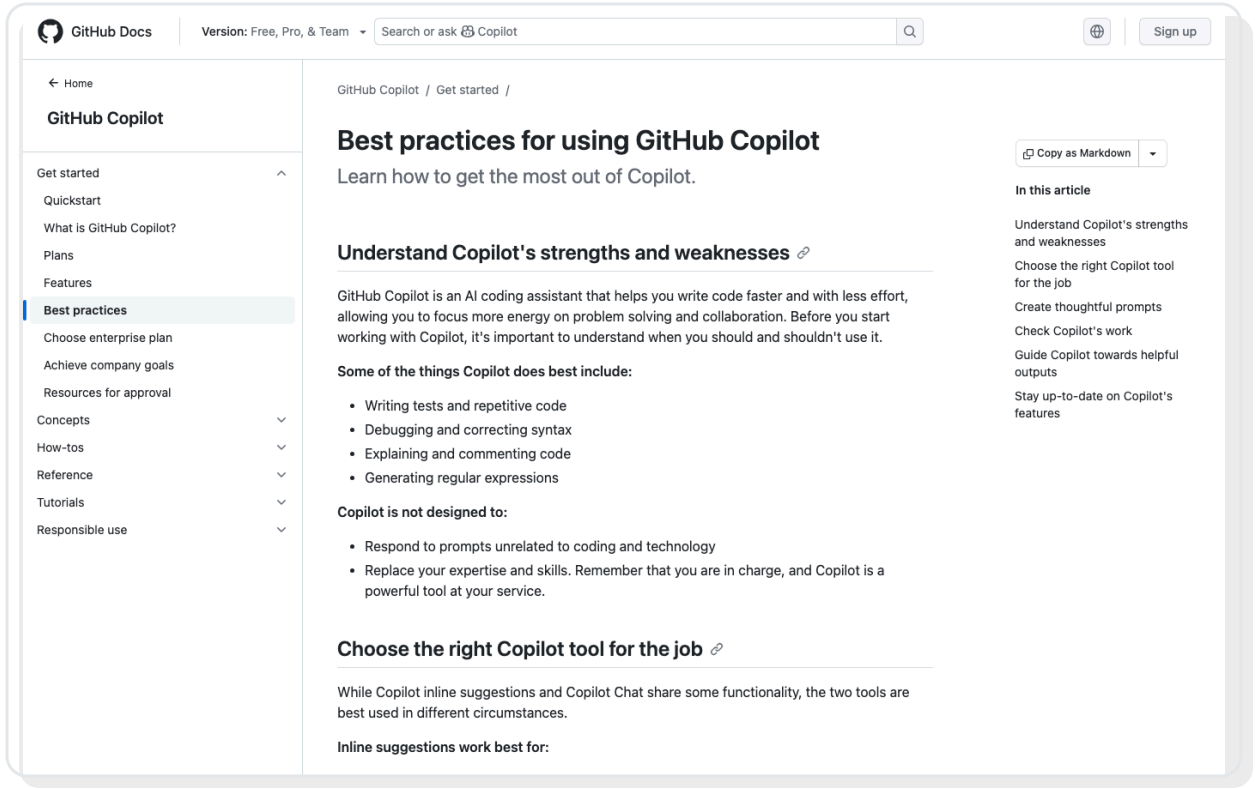
1件ずつ読んで承認する

追加された行と削除された行を目で追い、意図と合っているものだけ承認します。読んで承認した行の責任は、こちら側に移ります。

補足

公式が置いている前提

提供元の公式資料も、文脈を渡して小さく頼むことを前提にしています



GitHub公式のベストプラクティス。文脈を渡すこと、依頼を小さく分けること、出力を必ず検証することが繰り返し書かれています

判断は手放さない

速く書けるようになるほど、読む力と切る判断が効いてきます

> **候補を出すのはAI**
> **採用を決めるのは人**
>
> 客先の環境で動くかどうかは、AIには見えていません。動作確認と受け入れの判断は、今日以降も自分の側に残ります。

クイズ

理解度チェック 1

Q. 複数のファイルにまたがる修正を一度に任せたいとき、最も適した使い方はどれですか。

- **A** 補完に任せて、カーソル位置から順に書かせる
- **B** チャットに1ファイルずつ内容を貼って相談する
- **C** エージェントに、触ってよい範囲を指定して依頼する
- **D** モデルを切り替えてから補完を使う

正解は次のページで確認します。まず自分で考えてみてください。

解説

解説

正解はC。複数ファイルの変更は、範囲を指定してエージェントに渡します

C **エージェントに、触ってよい範囲を指定して依頼する**

エージェントは複数ファイルの編集とやり直しをまとめて行い、変更点を差分で返します。触ってよい範囲を先に伝えるほど、意図から外れません。出てきた差分を読んで承認するかどうかは、人が決めます。

A 補完に任せて、カーソル位置から順に書かせる

補完はカーソル位置の続きを数行出す機能で、開いていない他のファイルには手が届きません。

B チャットに1ファイルずつ内容を貼って相談する

進みはしますが、ファイル間の整合を人が取り直すことになり、修正が増えるほど破綻します。

D モデルを切り替えてから補完を使う

モデルを変えても補完が扱う範囲は変わりません。範囲を決めるのは機能の側です。

午後は、この3つを実際に切り替えながら触ります。

午前の持ち帰り

午前で手元に残ったものと、言葉にできるようになったものを確認します

動く環境	VSCodeでチャットが日本語で返り、エディタに補完のグレー文字が出る状態。今日この後の全部がこれを前提にしています。
3つの使い分け	補完・チャット・エージェントを、機能名ではなく任せる作業の大きさで説明できる状態。
承認の責任	エージェントが出した差分を読んで承認した行は、自分が書いた行として扱われます。ここは人が持ちます。

疎通確認が残っている人は、午後の演習中に個別で見ます。止まったままにしないでください。

午前から午後へ

ここから70分は、画面を見る時間ではなく**指を動かす時間**です。

ここまでにやったこと

午前の最後に、動く環境と3つの使い分けと承認の責任の3つを確認しました。昼をはさんで、ここからその3つを手で動かします。午前に手を動かしたのはセットアップの確認だけでした。

これからやること

ここから70分で、Copilotの入口3つを全部自分の指で触ります。配布の作業記録アプリを読ませ、スラッシュコマンドを1つ実行し、エージェントモードの切替場所を覚えます。

そのあとどこで効くか

午後の後半では、仕様を書いてWebアプリを生成します。そこで使う操作は、この70分に出てくるものだけです。切替の場所とファイルの承認をここでおさえておくと、後半で手が止まりません。

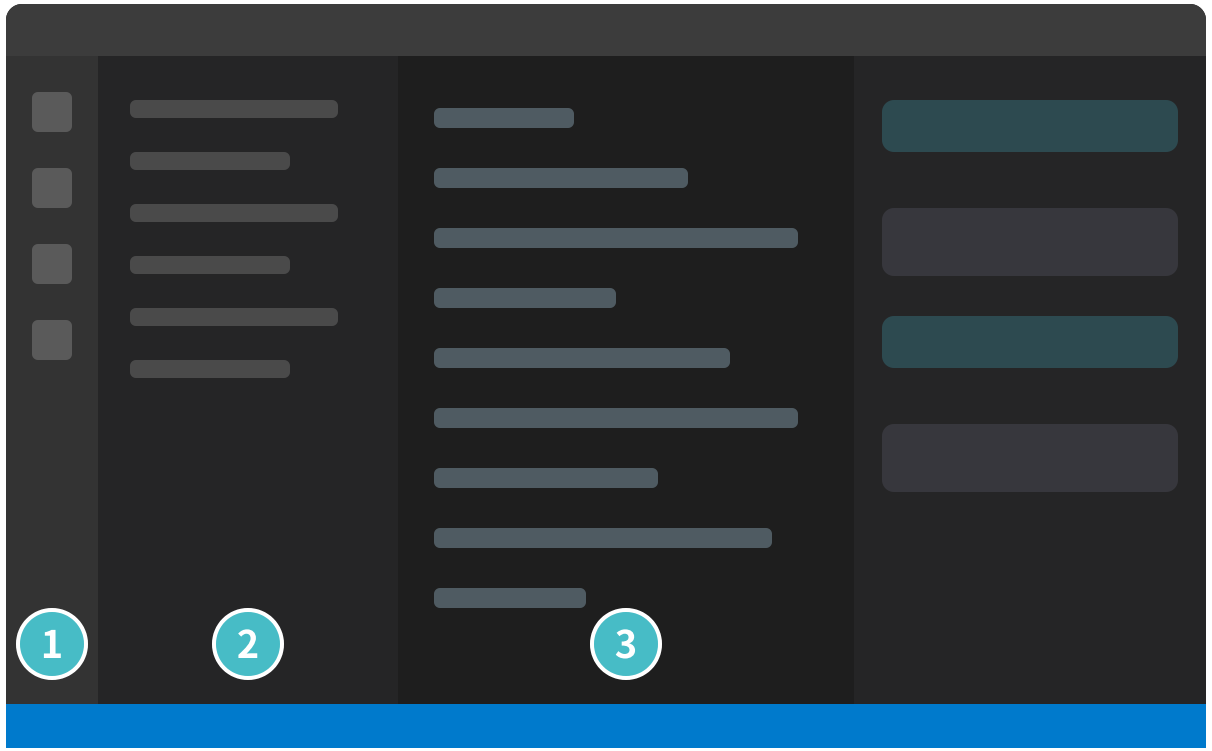
03

Copilotの基本操作

チャット・スラッシュコマンド・エージェントモードを自分の指で動かせる状態にします

Copilotが出る3か所

午前に名前をそろえた5か所のうち、Copilotが顔を出すのは3か所です。

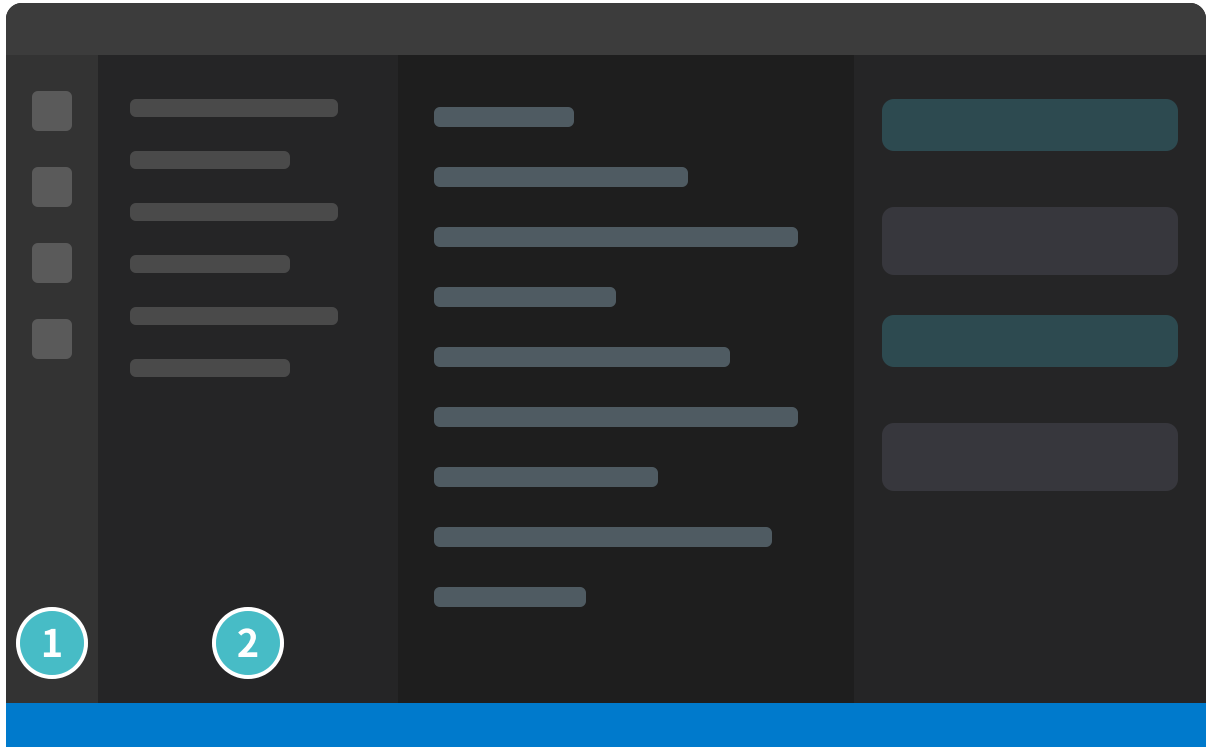


1 **アクティビティバー**
左端の縦一列。今日押すのはエクスプローラーとチャットの2つのアイコンだけです

2 **エクスプローラー**
午前に開いた tpro-work の中身。作業記録アプリと spec5.txt が並んでいれば、開いているフォルダは合っています

3 **エディタ**
Copilotが出る1か所目。書きかけの行の続きがグレーの文字で出たら、それがインライン補完です

Copilotが出る3か所（続き）



1 チャットパネル
2か所目で今日の主役。日本語で書いて送ります。入力欄の下に Ask / Edit / Agent の切替があります

2 ステータスバー
3か所目。Copilotのアイコンが出ます。斜線や警告の印が付いていたら、サインインか拡張の状態を疑います

実演

最初に送る1文

日本語のまま送れます。まず全員で同じ1文を送ります。

チャットパネルの入力欄に、次の2行をそのまま打って送ります。Enterで送信、改行は Shift+Enter です。長い文を貼るときに途中で送信されて焦る人が出るので、先に覚えてください。

```
いま開いているフォルダに何のファイルがありますか。
日本語で答えてください。

英語で返ってきた人は、続けてこの1行だけ送ります。
日本語で答えて。
```

> 返答の内容の正確さは、ここでは気にしません。日本語のテキストが返ってくれば通っています。

3つの入口は、それぞれ別の場所に出ます。切替は入力欄の下です。

インライン補完	エディタに書きかけた行の続きが、グレーの文字で出ます。Tab で受け入れ、Esc で捨てます。今開いているファイルの続きしか書けません
チャット	右のチャットパネルで質問します。返ってくるのは文章とコード片だけで、ファイルは書き換わりません。使うかどうかは自分で決めて、コピーして貼ります
エージェントモード	チャット入力欄の下にある Ask / Edit / Agent のドロップダウンで Agent を選びます。切替の場所はここだけです

ドロップダウンの表示位置はバージョンで変わります。見当たらないときは、チャット入力欄の周りを上下とも探してください。

スラッシュコマンド

スラッシュコマンドは、毎回同じ指示文を書く手間を省きます。

入力欄で / を打つと、使えるコマンドの一覧が出ます。名前を暗記する必要はありません。よく使う4つを、打つ文字そのままで示します。

```
/explain 選択した範囲の動きを説明させる
/fix エラーの原因と直し方を出させる
/tests 選択部分のテスト案を出させる
/clear 会話履歴を捨てて文脈をリセットする

実際の送り方。先にコードを範囲選択してから、この1行を送ります。
/explain この関数が何をしているか、初めて読む人向けに3行で説明してください
```

> コマンドの顔ぶれはバージョンで増減します。一覧に無いものがあっても、同じ内容を日本語で書けば同じ結果になります。

同じ指示を出しても、ファイルが変わるかどうかが違います。

チャット

返ってくるのは文章とコード片です。
エクスプローラーのファイル一覧は何も変わりません。
使うかどうかを自分で判断し、コピーして貼ります。
貼る場所を間違えるのは自分の責任で、ファイルは勝手に増えません。

エージェントモード

ファイルの新規作成と書き換えまで実行します。
エクスプローラーに新しいファイルが出て、変更は差分で提示されます。承認するまで書き込まれません。
承認したあとは、自分が書いたコードとして扱われます。

補足

差分の承認と戻し方

読まずに承認した時点で、それは自分が書いたコードになります。

1. **差分が出る**
   変更される予定が、追加される行と削除される行に分けて並びます。この時点でファイルはまだ変わっていません
2. **差分を読む**
   どのファイルの何が変わるかを見ます。ファイル名が想定と違っていたら、そこで止めて指示文を直します
3. **承認して確定する**
   承認のボタンを押すと、ファイルに書き込まれます。ボタンの表示名はバージョンで変わるので、当日の画面で確認します
4. **戻す**
   Ctrl+Z で直前の変更を戻すか、チャット側の変更を元に戻すを使います。今日はGitを使わないので、退路はこの2つだけです

ワーク

演習 入口の一巡

配布アプリを読ませ、切替を1回やって、3つの入口を全部触ります。

どんな場面か

客先で最初にやることは、たいてい引き継いだ画面の中身を読むところからです。中身が分からないまま直すと事故ります。ここでは配布の作業記録アプリを Copilot に読ませ、自分の言葉で説明できる状態にします。あわせて、後半で使うエージェントモードの切替場所を指で覚えます。

やること

1. 午前に開いた tpro-work の中の 作業記録アプリ フォルダを開き、index.html をエディタで開く
2. script タグの中にある render という関数を、行ごと範囲選択する
3. チャット入力欄に「/explain この関数が何をしているか、初めて読む人向けに3行で説明してください」と打って Enter で送る

ワーク

演習 入口の一巡（続き）

やること

1. 返ってきた説明を読み、自分の理解と違う箇所か、説明が足りないと感じた箇所を1つ、tpro-work フォルダの中に notes.txt を新規作成して1行で書く
2. チャット入力欄の下の Ask / Edit / Agent のドロップダウンを開いて Agent を選び、表示が Agent に変わったのを見てから Ask に戻す。この手順では送信しません

できあがるもの

デスクトップの tpro-work フォルダの中に notes.txt があり、1行以上書かれている状態。他のファイルは作りません。index.html は書き換えません。

演習1の達成基準

3つ全部が満たせていれば、午後の後半に必要な操作は揃っています。

達成の目安

- /explain の返答が日本語で、render 関数の話になっている。表を作り直して並べ直しているという趣旨が読み取れる
- tpro-work フォルダの中の notes.txt に、自分の理解と違った点か、説明が足りないと感じた点が1行以上書けている
- チャット入力欄の下のドロップダウンの表示が Agent に変わり、そのあと Ask に戻せた

つまずいたら

- コマンドの一覧に /explain が出ない。バージョンで顔ぶれが変わります。選択したまま「選択した部分を3行で説明してください」と日本語で書けば同じ結果になります
- ファイル全体の説明が返ってきた。範囲選択が外れています。選び直して /clear を1回送り、もう一度送ってください
- ドロップダウンが見つからない。入力欄の上下を探してください。位置はバージョンで変わります。見つからない人は手を挙げてください
- 返答が英語で返る。続けて「日本語で答えて」と送ってください。会話は引き継がれます
- index.html を書き換えてしまった。Ctrl+Z で戻してください。このファイルは後半の演習で使います

演習1の達成基準（続き）

時間が余ったら

- 同じ render 関数にもう一度 /explain を送り、1回目と文面が違うことを確かめる。文が変わっても中身が同じかどうかを見る
- 同じ選択範囲で /tests を送り、返ってきたテスト案の中に「1件も入力していないとき」が入っているかだけ見る。実装はしません
- index.html の script の中に // 工数の合計を返す関数 とコメントを書いて改行し、インライン補完のグレー文字が出るのを見る。Esc で捨てて、ファイルは書き換えない

クイズ

理解度チェック 2

Q. 範囲を選ばずに /explain を送ったところ、ファイル全体の説明が返ってきました。関数1つだけの説明が欲しいとき、まずすることはどれですか。

- **A** その関数を行ごと範囲選択して、もう一度 /explain を送る
- **B** 「3行で短く」と書き足して、もう一度送る
- **C** Agent に切り替えて、同じ文を送る
- **D** その関数だけを新しいファイルにコピーして、そのファイルを開いて送る

正解は次のページで確認します。まず自分で考えてみてください。

解説

解説

正解はA。説明の対象を決めているのは、指示文の書き方ではなく渡した範囲です。

A **その関数を行ごと範囲選択して、もう一度 /explain を送る**

関数を行ごと範囲選択してから送ると、その範囲だけが対象になります。選択していないときは、ファイル全体か直前の会話が対象になります。選び直しても前の説明を引きずるときは、/clear を1回送ってから送り直します。

B 「3行で短く」と書き足す

短くはなります。ただし対象はファイル全体のままなので、全体の要約が3行で返ってくるだけで、聞きたい関数の話にはなりません

C Agent に切り替えて同じ文を送る

Agent はファイルを書き換える入口です。読みたいだけの場面で選ぶと、頼んでいない修正まで提案されます

D その関数だけ別ファイルにコピーして送る

動きはします。ただし切り出す手間がかかり、呼び出し元が見えなくなって説明が浅くなります。選択するだけで足ります

噛み合わなくなったら /clear を1回送り、選び直してからもう一度送ります。

操作から書き方へ

ルールを先に決めます。型を先に教えると、練習の1文に客先名が入ります。

ここまでにやったこと

ここまでの70分で、チャット、スラッシュコマンド、エージェントモードを一巡しました。差分は承認するまでファイルに書き込まれないことも、実際の画面で確認しました。操作はこれで一通り出ています。

これからやること

ここから50分は、書き方とルールの時間です。先に客先情報の丸め方を決めて、そのあとに指示文の型を作ります。順番を逆にすると、練習で書いた1文に客先名が入ります。

そのあとどこで効くか

この型は、次のブロックで仕様からWebアプリを作るときに、そのまま貼って使います。動かなくなったときの相談文にも同じ型を使います。持ち帰るものはこの50分で作る1ファイルです。

04

プロンプト設計とガバナンス

客先情報の丸め方を決め、そのまま使える指示文の型を1つ持ち帰ります

客先情報の丸め方

客先の情報は、丸めてから入力します。最後に確認するのは自分です。

ルール	丸め方の実例
客先名・製品名・プロジェクト名・取引先名は業界・工程レベルに丸める	「A社の受注管理システム」を「流通業の基幹システム」に。判断の基準は、その語で検索して客先が特定できるかどうか
実データ・実ファイル・スクリーンショットは使用しない	本番のCSVは貼らない。列の意味だけ同じにしたダミーを自分で作って渡す
数値や仕様は具体値を避け、目的・課題・期待効果の粒度で記載する	「1日3万件を30分で処理」を「日次の処理件数が多く、処理時間が課題」に
提出前に受講者本人が社外に出ても問題ない内容かを確認する	送信ボタンを押す前に自分で1回読み返す。誰かが後で止めてくれる前提を持たない

> 今日の演習は全部ダミー題材なので、この場ではこのルールを踏みません。危ないのは明日から自席に戻ってからです。

クイズ

理解度チェック 3

Q. 客先の受注管理システムで起きた不具合を Copilot に相談したいとします。最も適切なのはどれですか。

A　エラーログをそのまま貼って、原因を聞く

B　流通業の基幹システムと言い換え、再現する最小のダミーコードを添えて聞く

C　客先名だけ伏せ、テーブル名と列名はそのまま貼って聞く

D　社内チャットで許可を取ってから、画面のスクリーンショットを貼って聞く

正解は次のページで確認します。まず自分で考えてみてください。

解説

解説

正解はB。構造だけ同じダミーに置き換えれば、丸めても再現できます。

B

流通業の基幹システムと言い換え、再現する最小のダミーコードを添えて聞く

業界と工程のレベルに言い換えたうえで、再現する最小のダミーコードを自分で作って渡します。1項目目の丸め方と2項目目の実データ不使用を同時に満たし、しかも相談の中身は薄くなりません。ダミーを作る手間は、列の意味だけ残して名前を変える数分です。

A

エラーログをそのまま貼る

ログにはファイルパス、ホスト名、内部のID体系が入ります。客先名を書いていなくても、パスから組織が分かることがあります

C

客先名だけ伏せて、テーブル名と列名はそのまま貼る

テーブル名や列名は業務の言葉でできています。その語で検索して客先が特定できるなら、伏せたことになりません

D

許可を取ってからスクリーンショットを貼る

スクリーンショットは使用しないと決まっています。許可の有無で例外にはなりません。画面に写る範囲は自分で制御しきれません

迷ったら、その語で検索して客先が特定できるかで判断します。特定できるなら丸めます。

プロンプトの4要素

4要素のうち、制約と出力形式が結果を大きく変えます。

役割	誰として答えるかを決めます。「JavaScriptのコードレビュアーとして」など。ここを凝っても効果は限られます
文脈	対象と動かす環境を書きます。「単一のHTMLファイル、ブラウザで直接開いて実行」など。今日の環境はこの1行で表せます
制約	やらないことを書きます。「外部ライブラリは追加しない」。これを書かないと、今日の環境では動かないコードが返ります
出力形式	返し方を指定します。「変更した行だけを差分で」。指定しないとファイル全体が丸ごと返ってきて、どこが変わったか読めません

4つの順番は覚えなくてよいです。送る前に「制約と出力形式を書いたか」だけを自問してください。

悪い例と良い例

原因を自分で断定して書くと、AIはそこ以外を見なくなります。

そのまま聞いた例

「合計がおかしいので直して」。返ってくるのは、ありそうな原因の一般論です。どのファイルの話かも、どう返してほしいかも書いていないので、こちらの環境では動かないコードが混ざります。「reduceがおかしいので直して」のように原因を断定して書くのも同じで、そこ以外を見なくなります。

4要素を入れた例

環境（単一のHTMLをブラウザで直接開いて実行）、現象（工数の合計欄に数字が足し算されずに並ぶ）、制約（外部ライブラリを追加しない）、出力形式（原因の候補を3つ、そのあとに修正した行だけ）を書きます。原因は書かず、見えている事実だけ渡します。全文は次のページに載せます。

補足

そのまま使える型

型を1つ持っておくと、指示の書き出しで迷いません。

前ページの良い例を、そのまま送れる全文で示します。この5行をデスクトップの tpro-work フォルダの中に prompt.txt という名前で保存してください。今日はこのファイルに指示文を足していきます。

```
あなたはJavaScriptのコードレビュアーです。
対象は単一のHTMLファイルで、ブラウザで直接開いて動かしています。
現象は、工数の合計欄に数字が足し算されずに並んで表示されることです。
制約は、外部ライブラリを追加しないことです。
出力は、原因の候補を3つ、そのあとに修正した行だけの順で返してください。

差し替えるのは5か所です。役割 / 対象と環境 / 現象 / 制約 / 出力の形
現象の行だけ毎回書き換わります。他の4行はほぼ固定で使えます。
```

> 使わない行は消してかまいません。役割の行を凝るより、制約の行を1つ足す方が効きます。

ワーク

演習 型の書き換え

自分の作業に置き換えて、丸めた状態で1回書きます。送りません。

どんな場面か

明日から自席で使うときに一番危ないのは、急いでいるときに客先名やテーブル名をそのまま貼ることです。落ち着いている今のうちに、丸めた状態で書く練習を1回だけしておきます。書いたものは送りませんし、提出もありません。

やること

1. デスクトップの tpro-work フォルダの中の prompt.txt を開く。前のページで保存したファイルです。作っていない人はここで新規作成して、投影の5行を写す
2. 5行の中身を、自分がいま関わっている作業に置き換えて書き換える。客先名・製品名・プロジェクト名・取引先名は、業界と工程の言葉に丸める
3. 書けたら自分で1回読み返し、その語で検索して客先が特定できる語が残っていないかを確認する
4. 残っていたら丸めた語に書き換えて、保存する。送信はしません

できあがるもの

デスクトップの tpro-work フォルダの中の prompt.txt に、5行が自分の言葉で埋まっている状態。送信も提出もありません。手元に残すだけです。

演習2の達成基準

制約と出力形式が埋まっていて、固有名詞が残っていなければ達成です。

達成の目安

- 5行のうち、制約の行と出力形式の行が空になっていない
- 社名・製品名・システム名・人名といった固有名詞が1つも残っていない。その語で検索して客先が特定できないことを自分で確認できている
- 現象の行が、原因ではなく見えている事実で書けている。「reduceがおかしい」ではなく「合計欄に数字が並んで出る」の形になっている

つまずいたら

- 何を書けばいいか浮かばない。直近1週間で誰かに聞いた質問を1つ思い出して、それを書いてください。題材を探すのに時間を使わないでください
- 丸めると相談として成立しない気がする。構造だけ同じダミーで足ります。列の意味は残して、名前だけ変えてください
- 制約が思いつかない。今日の環境の制約をそのまま書いてください。外部ライブラリを追加しない、単一のHTMLファイルで完結させる
- prompt.txt が見つからない。前のページで作っていない人はここで新規作成してください。デスクトップの tpro-work フォルダの中です

演習2の達成基準（続き）

時間が余ったら

- 同じ依頼を、制約の行を消した版ともう1つ書いて並べる。どちらを送りたくなるかを自分で見る
- 現象の行だけを3通り書き分ける。原因を断定した書き方、事実だけの書き方、事実に再現手順を足した書き方。3つ目が一番強いことを確かめる
- 書いたものを送るのは自席に戻ってからにする。送る前に、社外に出しても問題ない内容かをもう一度読み返す

ここまでとこれから

ここからは、聞く時間より手を動かす時間が長くなります。

ここまでにやったこと

午後前半で3つの入口を触り、客先情報の丸め方を決めて、指示文の型を prompt.txt に1つ書きました。ここまでは1往復ずつの練習で、送っていない指示文もあります。

これからやること

これから80分で、日本語の仕様5行から作業記録アプリを1本作り、ブラウザで動かします。生成そのものは数分で終わります。残りの時間は、動いたかどうかを自分で確かめることに使います。

そのあとどこで効くか

そのあと70分で、そのアプリに機能を足し、コードを1行だけ壊し、AIに直させます。ここまでやると、Day2で配布する既存アプリに手を入れるとき、読ませてから直させる順番が自分の手癖になります。

05

仕様5行から動くWebアプリ

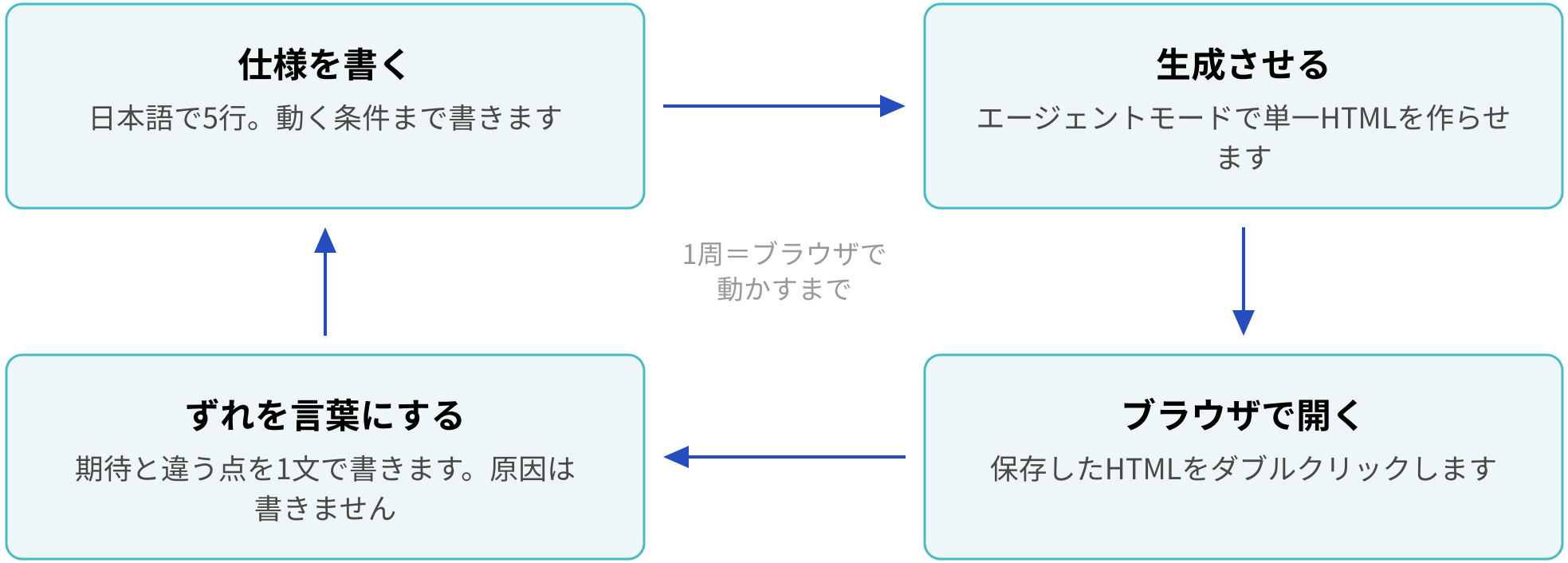
日本語5行を貼って、作業記録アプリをブラウザで動かせる状態にします

Day1の現在地

残りは150分。うち手を動かす時間が130分です。

生成と確認の1周

ブラウザで開くまでを1周と数えます。開かずに次を頼まないでください。



実行までの4手順

書く場所はVSCode、動かす場所はブラウザです。サーバは立てません。

1. **VSCodeで書く**
   エディタでHTMLを編集します。生成させた場合も、変更はエディタに反映されます
2. **保存する**
   Ctrl+S を押します。タブのファイル名の横の点が消えれば保存できています
3. **エクスプローラーで開く**
   Windowsのエクスプローラーでデスクトップの tpro-work フォルダを開きます。VSCode側でファイルを右クリックして エクスプローラーで表示 を選ぶと確実です
4. **ダブルクリックする**
   worklog.html をダブルクリックすると、既定のブラウザで開きます。アドレスバーには file:/// から始まるパスが出ます

> file:/// で開いているため、JavaScriptから外部ファイルを読み込む処理は動きません。Day2でCSVやExcelを読むときは、画面へのドラッグ＆ドロップで渡します。

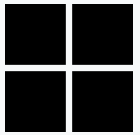
解説

書く場所と動かす場所

同じHTMLを、VSCodeで書いてブラウザで開きます

VSCode

worklog.html を編集して
Ctrl+S で保存します



Microsoft Edge

保存したファイルを
ダブルクリックして開きます。アドレスは file:///
から始まります

補足

tpro-work の中身

今日つくるファイルは、すべてデスクトップの tpro-work の中に置きます。

ファイル	中身	今日の使い方
spec5.txt	このあと貼る仕様5行だけが書かれたテキスト	全員のコピー元にします。手打ちしません
worklog-complete.html	作業記録アプリの完成版	生成がうまくいかなかった人の差し替え用です
prompt.txt	午後前半に作った指示文の型	今日送った指示と返答を追記していきます
作業記録アプリ／index.html	配布の既存アプリ	午後前半に /explain で読ませました。今日はもう開きません
lib フォルダ	表とグラフのライブラリ3本、合計約1.1MB	Day2以降で使います。今日は触りません

実演

仕様5行の実物

5行目が、ブラウザで動く形を決めています。

tpro-work/spec5.txt をそのままコピーして、チャットに貼ります。書き換えずに使ってください。

```
1. 日付・担当・工程・作業内容・作業時間（分）を入力するフォームを1つ置く
2. 「追加」ボタンで、入力内容を下の一覧表に1行追加する
3. 一覧表の下に、作業時間の合計を分単位で表示する
4. 未入力の項目があるときは追加せず、フォームの下に「未入力の項目があります」と表示する
5. HTML・CSS・JavaScriptを1つのHTMLファイルにまとめ、ブラウザで開くだけで動くようにする。ファイル名は worklog.html にする
```

5行と画面の対応

5行のそれぞれが、画面のどこか1か所に必ず出ます。

1行目	画面の上のフォーム。日付、担当、工程、作業内容、作業時間の5つの入力欄になります
2行目	追加ボタンと、その下の一覧表。押すと表に1行増えます
3行目	一覧表の下の合計欄。分の数値で出ます
4行目	フォームの下のメッセージ。空欄があるときだけ「未入力の項目があります」が出ます
5行目	ブラウザのアドレスバー。file:/// から始まるパスで開けていれば成功です

解説

差分と承認の場所

エージェントモードの変更は、承認するまでファイルに書かれません。

1. **アクティビティバー**
   チャットのアイコン。ここからチャットパネルを出します
2. **エクスプローラー**
   tpro-work の中身。生成に成功すると worklog.html がここに増えます
3. **エディタ**
   変更が差分で出ます。追加される行と削除される行が色分けされます
4. **チャットパネル**
   入力欄の下の Ask / Edit / Agent。今日は Agent を選びます。差分を確定するボタンは入力欄の上か差分の上に出ます
5. **ステータスバー**
   開いているファイルの種類。HTML と出ていれば、保存先を間違えていません

完成の判定4か所

見た目は人によって違います。この4か所が動いていれば完成です。

見る場所	入れる値	出るはずの表示
フォーム	日付、担当、工程、作業内容を埋め、作業時間に 30	追加ボタンを押せる状態になります
一覧表	30分、15分、45分の3件を続けて追加	表が3行になります
合計欄	上の3件を入れた状態	90 と出ます
メッセージ	どれか1つを空欄にして追加を押す	行は増えず「未入力の項目があります」が出ます

ワーク

演習 作業記録アプリ

5行を貼って、ブラウザで動くところまでを自分の手で通します。

どんな場面か

客先で口頭でもらった要件を、その日のうちに動く形で見せられると話が早く進みます。まずは自分の手元で、日本語の仕様から動くものが出るまでの所要時間を体感します。

やること

1. VSCodeで tpro-work フォルダを開き、チャットパネルの入力欄の下で Agent を選ぶ
2. tpro-work/spec5.txt を開き、Ctrl+A、Ctrl+C で5行をコピーする
3. チャットの入力欄に貼り付けて送信する
4. 提示された差分を上から読み、読んだうえで承認する
5. エクスプローラーでデスクトップの tpro-work を開き、worklog.html をダブルクリックする
6. 30分、15分、45分の3件を入力して、合計欄と一覧表を見る

できあがるもの

デスクトップの tpro-work フォルダの中に worklog.html ができます。ファイル名が違う場合は、チャットに「ファイル名を worklog.html にしてください」と送って直させます。

補足

達成基準 作業記録アプリ

3つ全部を自分で確かめた人だけが、次のブロックに進めます。

達成の目安

- worklog.html をダブルクリックすると既定のブラウザで開き、アドレスバーが file:/// から始まっている
- 30分、15分、45分の3件を追加すると一覧表が3行になり、合計欄に 90 と出る
- どれか1つの項目を空欄にして追加を押すと、行が増えず「未入力の項目があります」が出る

つまずいたら

- 画面が真っ白。F12のConsoleを見て、外部ファイルの読み込み失敗が出ていたら「外部ライブラリを使わず1つのHTMLだけで動くように直して」と送ります
- ダブルクリックしてもブラウザではなくVSCodeが開く。ファイルを右クリックして プログラムから開く でブラウザを選びます
- 合計欄が 3015 のように数字の連結になる。今日の後半で扱う現象と同じです。ここでは「合計は数値として足してください」と1文送って直します
- 空欄チェックが作業時間だけ効かない。3つの入力欄それぞれで試して、効かない項目を名指しで伝えます

補足

達成基準 作業記録アプリ（続き）

時間が余ったら

- 「合計を時間単位でも併記してください」と1文だけ追加で送り、変更された行だけを読みます
- 「削除ボタンを各行に足してください」と送り、追加後に3件の合計がずれないかを確かめます
- ページを再読み込みすると入力が消えることを確認します。仕様に書いていないので消えるのが正しい動きです

ここまでと次の70分

動いているものを、これからわざと壊します。壊し方は指定します。

ここまでにやったこと

5行の仕様から worklog.html が1本でき、3件入れて合計90が出るところまで来ました。生成にかかった時間は数分で、残りは確認と手直しでした。

これからやること

これから70分で、そのアプリに色付けの機能を1つ足し、合計を出しているコードを1行だけ壊し、症状だけを伝えてAIに直させます。壊すのは1行で、元の1行は資料に載っています。

そのあとどこで効くか

エラーが出ない不具合を症状の言葉だけで伝えて直させる、この往復がDay2の既存アプリへの機能追加でそのまま使えます。Day3のテスト設計でも、期待と実際のずれを言葉にする作業が中心になります。

06

機能追加とデバグ

記号1つで変わる挙動を自分の目で見て、症状の言葉だけでAIに直させます

実演

機能追加の指示文

既存の機能を変えないでください、を書き添えます。

worklog.html をVSCodeで開いた状態で、Agentのまま送ります。1回の指示で頼むのは1つだけです。

```
tpro-work/worklog.html を対象にしてください。
作業時間が60分以上の行の背景を、薄く色付けしてください。
既存の合計表示と未入力チェックは変更しないでください。
変更した行だけを示してください。
```

> 3行目が制約、4行目が出力形式です。午後前半に扱った4要素が、そのままこの4行に入っています。

ワーク

演習 機能追加

機能を足したら、前の機能が生きているかを毎回見ます。

どんな場面か

既存のコードに手を入れる仕事は、新しく作るより回数が多いです。レビューで本当に見るのは、AIが書いた新機能ではなく、触っていないはずの既存機能の方です。

やること

1. エクスプローラーで tpro-work/worklog.html をコピーして貼り付け、worklog-backup.html という名前で残す
2. 60分ちょうどの行に色が付くかどうかを、送る前にメモに書く
3. worklog.html を開いた状態で、前ページの4行をチャットに送る
4. 差分を読んで承認し、Ctrl+S で保存してブラウザを再読み込みする
5. 作業時間が 59分、60分、61分の3件を入力して、色の付き方を見る

できあがるもの

デスクトップの tpro-work フォルダの worklog.html を上書きします。新しいファイルは作りません。同じフォルダに worklog-backup.html が1つ残っている状態になります。

補足

達成基準 機能追加

予想が外れた人も達成です。どちらで実装されたかを言えれば達成です。

達成の目安

- 59分の行に色が付かず、61分の行に色が付いている
- 60分の行は、色が付いたか付かなかったかを見て、生成されたコードが 60 以上 と 60 より大きい のどちらで書かれているかを自分の口で言える
- 色を付けたあとでも、3件の合計が正しく出て、空欄のまま追加を押すと「未入力の項目があります」が出る
- tpro-work に worklog-backup.html が残っている

つまずいたら

- 未入力チェックか合計が消えた。「変更した行だけを示してください。既存の合計表示と未入力チェックは変更しないでください」を付けて再送します
- 60分ちょうどの行を入れていないので差が見えない。59、60、61の3件を入れ直します
- 色が薄すぎて判別できない。「背景色をもう少し濃くしてください」と1文だけ送ります
- バックアップを取り忘れた。いまここで worklog.html をコピーして worklog-backup.html を作ります

補足

達成基準 機能追加（続き）

時間が余ったら

- 60 以上 と 60 より大きい のどちらだったかを1行メモし、仕様の日本語をどう書けば一意になるかを考えます
- 「作業内容で絞り込む検索欄を表の上に足してください」と送り、合計と色付けが壊れていないかを確かめます
- 絞り込んだ状態の合計が、表示されている行だけの合計になっているかを確かめます。気づいても直さないでください。直すのは最後の演習が終わってからにします

実演

コンマ1つの差

初期値のコンマ1つで、合計の型が変わります。エラーは出ません。

合計を出している1行から、末尾の , 0 だけを消します。元の1行はこのページに残しておきます。

```
// 元のコード（tpro-work/worklog.html の中）
const total = rows.reduce((sum, r) => sum + Number(r.minutes), 0);
// 壊したコード。末尾の , 0 だけを削除する
const total = rows.reduce((sum, r) => sum + Number(r.minutes));
// reduce を使っていない実装のときは、こちらで同じ現象になる
// let total = 0; を let total = ''; に変える
```

手順はこの4つです。worklog.html をVSCodeで開く、Ctrl+F で reduce を探す、末尾の , 0 を消す、Ctrl+S で保存する。

壊れた画面の見え方

画面には数字が出ます。だから気づきません。

, 0 を消したあと

30分と15分を入れると、合計欄が 合計: [object Object]15 になります。エラーダイアログは出ません。さらに、1件も入力していない状態で追加を押すと画面が更新されなくなり、F12のConsoleに TypeError が出ます。数字が出ているので、ぱっと見では気づけません。

, 0 があるとき

30分と15分を入れると、合計欄に 合計: 45 分と出ます。1件も入力していない状態では 合計: 0 分です。F12のConsoleにも何も出ません。判定できるのは、合計欄の値を自分で計算して見比べたときだけです。

ワーク

演習 1行だけの破壊

自分のファイルを、指定された1行だけ壊します。

どんな場面か

AIが返したコードを読まずに承認すると、この種の1文字違いが混ざります。混ざったときに画面がどう見えるかを一度自分の目で見ておくと、次に同じものを見たとき数秒で気づけます。

やること

1. tpro-work/worklog.html をVSCodeで開き、Ctrl+F で reduce を検索する
2. 合計を出している行の末尾の , 0 を消す。reduce が無い人は let total = 0 を let total = '' に変える
3. Ctrl+S で保存する
4. ブラウザに切り替えて再読み込みし、30分と15分の2件を入力する
5. 1件も入力していない状態にしてから追加を押し、F12 で Console タブを見る

できあがるもの

成果物はありません。合計欄に 合計: [object Object]15 が出ていれば達成です。tpro-work の worklog-backup.html は触らずに残しておきます。

補足

達成基準 破壊の確認

2つの症状を両方見た人だけが、次の修正演習に進みます。

達成の目安

- 30分と15分を入れると、合計欄に 合計: [object Object]15 と出る
- 1件も入力していない状態で追加を押すと画面が更新されず、F12 の Console に TypeError が出ている
- tpro-work に worklog-backup.html が残っている

つまずいたら

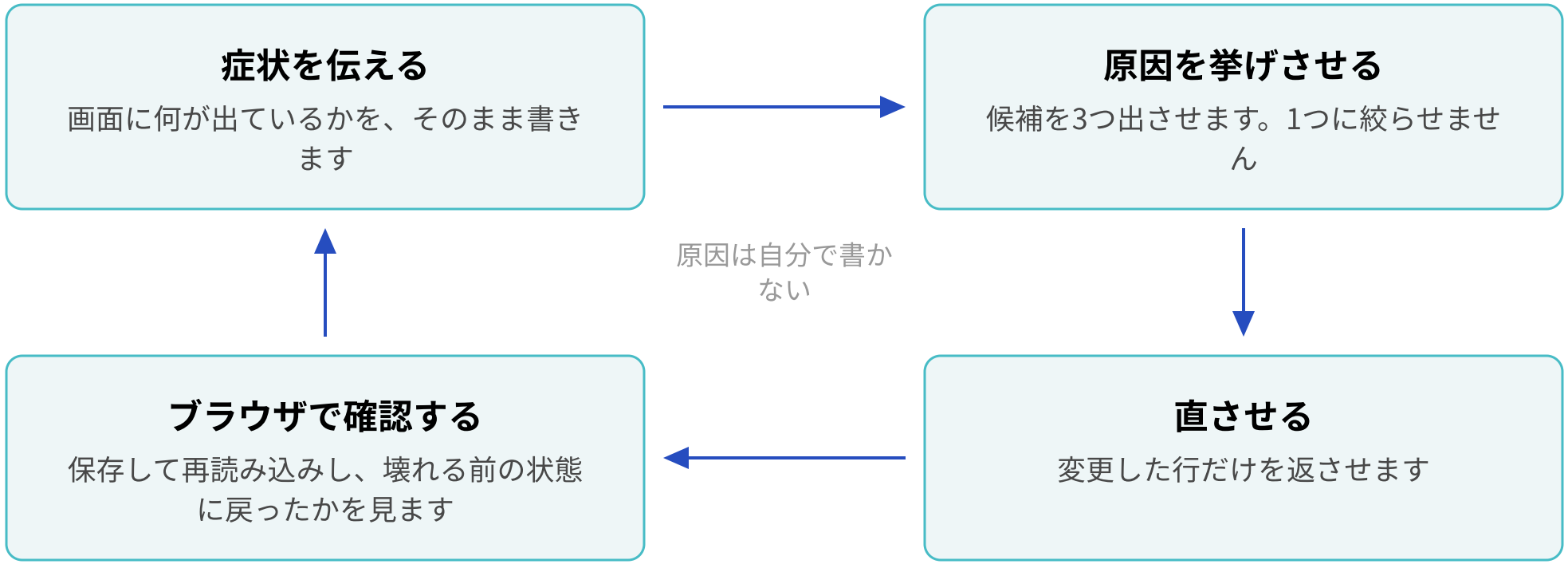
- 何も変わらない。Ctrl+S で保存してから、ブラウザを再読み込みします
- 画面が真っ白になった。括弧を消しすぎています。前のページの元の1行をそのまま打ち直します
- 合計が 45 のまま。reduce ではない実装なので、let total = 0 を let total = '' に変えます
- Console の場所が分からない。F12 を押して、上のタブの Console を選びます

時間が余ったら

- この1行以外に、同じくらい静かに壊せそうな場所を1つ見つけてメモします。実際には壊さないでください
- Console に出ている TypeError の文面を、そのままメモに写しておきます。次の演習で貼るかどうかを自分で決めます

デバグの4手

症状を伝える、原因を挙げさせる、直させる、ブラウザで確認する。



> 原因が分かっているときは先に書いた方が速いです。ただし推測が外れていると、AIも一緒に外れます。

株式会社ギブリー (Givery, Inc.)
〒150-0036 東京都渋谷区南平台町15-13 帝都渋谷ビル8F

実演

症状の伝え方

原因の推測を1文字も書かずに、現象だけを書きます。

壊した状態のまま、Agentのチャットにこの5行を送ります。tpro-work/prompt.txt に貼っておくと、次から使えます。

```
単一のHTMLファイルをブラウザで直接開いて動かしています。
作業時間に 30 と 15 を入れると、合計欄に 合計: [object Object]15 と表示されます。
1件も入力していない状態で追加を押すと、画面が更新されません。
原因の候補を3つ挙げ、そのあとに修正した行だけを示してください。
外部ライブラリは追加しないでください。
```

> 1行目が文脈、2行目と3行目が現象、4行目が出力形式、5行目が制約です。午後前半の4要素がそのまま入っています。

ワーク

演習 AIによる修正

症状だけを渡して直させ、直った理由を読みます。

どんな場面か

客先で受け取る不具合報告は、たいてい原因が書かれていません。画面に何が出ているかしか分からない状態から、原因の候補を出して潰していく。その最初の1往復をAIと一緒にやります。

やること

1. 壊した状態のまま、前ページの5行をチャットに送る
2. 返ってきた原因の候補3つを読み、どれが当たっていそうかを自分で1つ選ぶ
3. 提示された変更を承認し、Ctrl+S で保存してブラウザを再読み込みする
4. 30分と15分の2件を入れて合計が 45 になることを確認し、1件も入力していない状態でも 0 と出ることを確認する
5. 送った5行と、返ってきた原因説明を tpro-work/prompt.txt に追記して保存する

できあがるもの

デスクトップの tpro-work フォルダの worklog.html が壊れる前の状態に戻り、同じフォルダの prompt.txt に送った5行と原因説明が追記された状態になります。prompt.txt が無い人は、ここで新規作成します。

補足

達成基準 修正の確認

合計が数値に戻り、前に足した機能も残っている状態が達成です。

達成の目安

- 30分と15分の2件を入れると、合計欄に 45 と出る
- 1件も入力していない状態で追加を押すと、合計が 0 と表示され、Console に何も出ない
- 59分、60分、61分の色付けと、空欄のときの「未入力の項目があります」が残っている
- tpro-work/prompt.txt に、送った5行と返ってきた原因説明が入っている

つまずいたら

- 色付けや検索欄が消えた。「変更した行だけを示してください。既存の機能は変更しないでください」を付けて、消えた機能だけをもう一度足させます
- 合計が 0 ではなく空欄になる。「1件も入力していないときは 合計: 0 分 と表示してください」と1文送ります
- 何度送っても直らない。tpro-work/worklog-backup.html を worklog.html という名前でコピーして戻し、そこから1回だけやり直します
- prompt.txt を作っていない。VSCodeのエクスプローラーで tpro-work を右クリックして 新しいファイル で作ります

補足

達成基準 修正の確認（続き）

時間が余ったら

- 「reduce の初期値が抜けているので直してください」と原因を先に書いた場合に、返答がどう変わるかを比べます
- 60分以上の判定を 60 より大きい に書き換えて壊し、同じ手順で症状だけを伝えて直させます
- prompt.txt に、今日うまくいった指示文を3つだけ残して、他は消します。明日から使う形に整えます

補足

復旧チェック5項目

壊れたまま持ち帰らないでください。5項目を全員で確認します。

1. **開く**
   tpro-work/worklog.html をダブルクリックして、画面が表示されます
2. **3件入れる**
   30分、15分、45分を追加して、一覧表が3行になります
3. **合計を見る**
   合計欄に 90 と出ます
4. **空で押す**
   1件も入力していない状態で追加を押すと、合計が 0 と出ます
5. **色を見る**
   59分、60分、61分を入れて、色付けが動いています

1つでも欠けている人は手を挙げてください。tpro-work/worklog-complete.html を worklog.html という名前でコピーして差し替えます。

ここまでと振り返り

今日やったことは、全部1つの円の中に入っています。

ここまでにやったこと

5行の仕様から worklog.html を作り、機能を1つ足し、1行壊して、症状だけを伝えて直しました。ブラウザで開いて確かめた回数は、多い人で5回を超えています。

これからやること

これから20分で、今日の操作を1つの円に畳み直します。個別の操作を覚えて帰るのではなく、回し方を持ち帰ってもらいます。

そのあとどこで効くか

その円は、Day2で配布の既存アプリに機能を足すときも、Day3でテストケースを作るときも同じ形で使います。Day2の冒頭でこの円をもう一度出します。

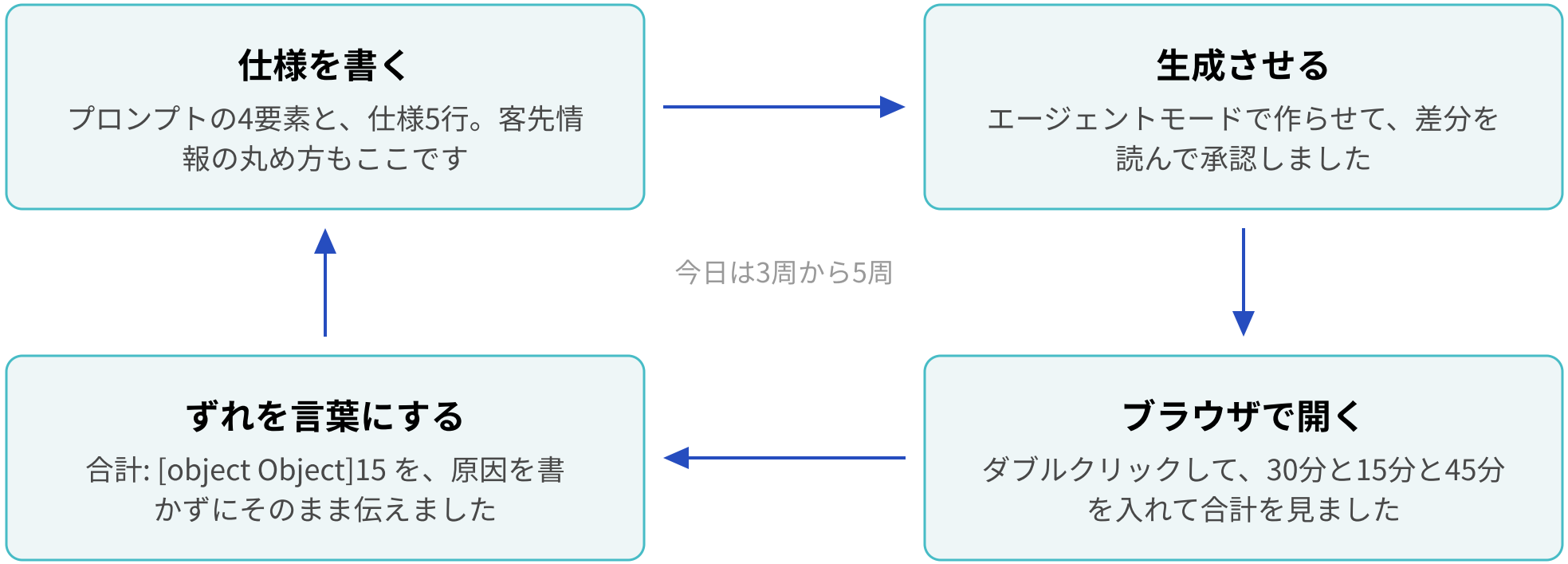
07

振り返りとDay2予告

今日の往復を1枚に畳み、次回までに残しておくものを確認します

今日の1周の中身

今日やった操作は、この4手のどこかに全部入っています。



今日の結論

生成の速さではなく、確かめる速さが成果を決めます。

> **生成は数分で終わりました**
> **確かめる10秒を毎回払えるかどうかです**
>
> 今日見つけたずれは全部、3件入れて合計欄を見るだけで見つかりました。特別な道具は使っていません

Day2の内容

Day2は、既存コードを読ませるところから始めます。

モデルの使い分け
用途別にモデルを切り替えます。今日は意図的に触れていません。午前に70分とっています

既存Webアプリへの機能追加
今日作ったものではなく、配布する既存のアプリを読ませてから手を入れます。tpro-work の中にあるアプリです

チームへの展開とテスト
前提を書いたファイルを置いて指示の書き方を共有します。午後はテスト観点の抽出とテストケース生成です

次回までの持ち物

tpro-work フォルダとサインイン状態を、消さずに残してください。

tpro-work フォルダ	worklog.html、prompt.txt、配布物を全部そのまま残します。Day2の冒頭で使います
Copilotのサインイン	次回まで約1週間空きます。サインアウトしないでください。ゼロからのセットアップはDay2ではやりません
自席での1回	今日の型を、自分の業務で1回だけ試してみてください。宿題ではありません。客先情報の丸め方を守れる題材でお願いします